MySQL: DataDefinitionLanguage

Prof. Viglietti Francesco

Www.in4matika.altervista.org

Impostazione della didattica dell'SQL

Il percorso didattico che che utilizzeremo per illustrare le funzionalità del linguaggio SQL è suddiviso in 4 fasi:

1. Creazione di Basi di dati, tabelle e indici
2. Inserimento, cancellazione e modifica dei dati
3. Interrogazioni di base
4. Interrogazioni avanzate

Ciascuna fase consiste in una spiegazione dettagliata di tutte le funzionalità del linguaggio attraverso la definizione di un database di esempio all'interno del DBMS MySQL. La spiegazione delle slide è accompagnata dalla interazione in tempo reale da parte degli studenti, che, al fine di fissare in maniera più efficace i concetti, replicheranno l'immissione di tutte le sequenze di comandi trattate.

Ciascuna fase si conclude con l'assegnamento di uno o più esercizi utili a fissare i concetti

Configurazione

L’immissione delle sequenze di comandi SQL esposti attraverso le slide verrà svolta all’interno dell’interfaccia a linea di comando (MySQL Monitor) di MySQL. Il tutto può essere trovato nel pacchetto completo EasyPhp, XAMPP o WAMP

E’ quindi necessaria l’installazione del tool es. XAMPP. Eseguire l’installazione completa. Impostare come password di root il singolo carattere x. Non attivare l’utente anonimo.

Lanciare XAMPP poi, eseguire il comando cmd da esegui, risalire quindi fino alla directory in cui è posizionato MySql e accedere alla cartella mysql e bin.

... \..\mysql\bin

Lanciare quindi il comando per entrare dentro mysql.

> mysql -u root

Creazione DB, tabelle e indici

Questo modulo è dedicato alla creazione delle strutture base di un DB. Creazione basi di dati, tabelle e indici, all'interno del DBMS open source MySQL.

In particolare, si parlerà di:

- creazione di una base di dati
- selezione di una base di dati
- creazione di tabelle
- colonne ed tipi di dato in MySQL
- eliminazione di tabelle, indici e basi di dati
- modifica della struttura delle tabelle.

Creazione DB, tabelle e indici

Utilizzeremo un esempio abbastanza semplice, la base di dati degli impiegati già vista precedentemente.

Lo schema della base di dati è:

Impiegato(IDimpiegato, nome, mansione, IDdip)

Dipartimento(IDdip, nomeDip)

CompetenzeImpiegato(IDimpiegato, competenza)

Cliente(IDcliente, nome, indirizzo, nomeReferente, telReferente)

Assegnamento(IDcliente, IDimpiegato, dataLavoro, oreEffettuate)

Creazione DB, tabelle e indici

Continueremo la trattazione inserendo man mano i comandi necessari per creare questa base di dati. Utilizzeremo:

- sempre il monitor, in modo da avere una maggiore comprensione di come sono fatte le varie strutture.
- i comandi di definizione dei dati del linguaggio SQL per creare le basi di dati, le tabelle e gli indici.

Come accennato in precedenza SQL è l'acronimo di Structured Query Language (linguaggio di interrogazione strutturato), un linguaggio usato per creare e interrogare le basi di dati relazionali.

L'SQL è formato da due parti ben distinte dal punto di vista semantico: un linguaggio per la definizione dei dati (DDL o Data Definition Language), adatto per creare le strutture delle basi di dati, e un linguaggio per l'interrogazione dei dati (DML o Data Manipulation Language).

Maiuscole e minuscole

Prima di iniziare a usare SQL e a creare gli identificatori in MySQL, è necessario parlare brevemente dell'uso di maiuscole e minuscole.
Le parole chiave di SQL non variano se usate lettere maiuscole o minuscole per scriverle. Questo è uno standard nel mondo delle basi di dati.
In MySQL, il fatto che maiuscole e minuscole siano importanti nei nomi delle basi di dati e delle tabelle dipende dal sistema operativo. Questo avviene perché, generalmente, ogni base di dati risiede in una propria directory e ciascuna tabella in un proprio file. I nomi di queste directory e di questi file seguono regole diverse a seconda del sistema operativo in cui si trovano.
In pratica, se usate Windows non importa se usate lettere maiuscole o minuscole per scrivere i nomi delle basi di dati e delle tabelle (cioè Impiegato, impiegato e IMPIEGATO indicano sempre la stessa tabella), mentre se usate un sistema basato su Unix la cosa cambia molto.

Maiuscole e minuscole

Questo può generare un po' di confusione, soprattutto se pensate che in MacOS X sono ammesse entrambe le possibilità: la prima preferendo l'opzione standard HES+, la seconda scegliendo invece UFS.
Una buona norma per evitare confusione è trattare sempre tutti gli identificatori come se maiuscole e minuscole fossero importanti, anche se state lavorando su un sistema Windows. Questa regola vi permetterà di spostarvi facilmente da una piattaforma all'altra. Inoltre, usare due forme diverse per lo stesso identificatore, ad esempio Impiegato e impiegato, farebbe confondere anche un eventuale utente che leggesse il codice; insomma è una brutta idea.
I nomi delle colonne, degli indici e gli alias, in MySQL non dipendono mai dall'uso di minuscole e maiuscole.

Identificatori in MySQL

Un identificatore è semplicemente il nome di un alias, di una base di dati, di una tabella, di una colonna oppure di un indice. E il modo in cui viene identificato l'oggetto. Prima di iniziare a creare una base di dati e le tabelle è bene vedere quali identificatori sono validi in MySQL.
In genere un identificatore può contenere qualsiasi carattere con le seguenti eccezioni:

- non sono mai ammesse le virgolette, cioè i caratteri ASCII(0) e ASCII (255);
- i nomi di base di dati possono contenere tutti i caratteri che si possono usare per un nome di directory eccetto i caratteri /, \ e . che hanno un significato speciale;
- i nomi di tabella possono contenere tutti i caratteri che si possono usare per un nome di file eccetto . e /.

Tutti gli identificatori, eccetto gli alias, possono avere fino ad un massimo di 64 caratteri. I nomi degli alias (che vedremo più avanti) possono contenere fino a 255 caratteri.

Identificatori in MySQL

Una strana regola degli identificatori in MySQL è il fatto che sono ammesse anche parole riservate a patto che siano messe tra virgolette. Ad esempio, potreste avere una tabella chiamata TABELLA. Naturalmente, il fatto che possiate usare parole chiave non vuol dire che sia una buona regola farlo e questa abitudine di solito va evitata.

Anche se la cosa non dovesse confondere voi, potrebbe mandare in confusione il programma mysqldump che è usato frequentemente per i backup.

Esiste una lista abbastanza breve di parole riservate che potete usare senza le virgolette come identificatori in MySQL.

Questa possibilità contrasta con lo standard ANSI per SQL, ma è molto comune nell'uso quotidiano. Gli esempi più comuni sono DATE e TIME usate come nomi di colonna.

Creazione di una base di dati

Dopo il progetto, il primo passo per creare una base di dati consiste, abbastanza logicamente, nel dire a MySQL che volete creare una base di dati. Lo facciamo col comando SQL create database, come segue:

create database impiegato;

Potete controllare se il comando ha funzionato digitando:

show databases;

Dovreste vedere nella lista delle basi di dati presenti nel sistema anche la base di dati impiegato
Adesso avete una base di dati vuota, che attende la creazione delle sue tabelle.

Selezione di una base di dati

Prima di creare le tabelle, o di fare qualsiasi altra cosa con la base di dati impiegato, occorre dire a MySQL che si vuole usarla. Per farlo si utilizza il comando ***use***:

use impiegato;

Adesso la base di dati impiegato è selezionata e tutto quello che farete in seguito agirà sempre su di essa.

Creazione delle tabelle

Per creare le tabelle nella base di dati usate il comando SQL create table. Di solito il comando ha questa forma:

create table nome_tabella (definizione della tabella) [type= tipo di tabella];

Significa che si inizia con le parole **create table**, poi si inserisce il nome che si vuole dare alla tabella e in seguito si introduce la definizione di un certo numero di colonne. In fondo al comando si può, facoltativamente, specificare che tipo di strumento di memorizzazione per la tabella si vuole utilizzare.

Creazione tabelle: Esempio

Vediamo un esempio. Il listato seguente mostra un insieme di comandi SQL che possono essere usati per creare la base di dati impiegato.

```
drop database if exists impiegato;
create database impiegato;
use impiegato;

create table dipartimento( IDdip int not null auto_increment primary key, nomeDip varchar(30));

create table impiegato( IDimpiegato int not null auto_increment primary key, nome varchar(80), mansione varchar(30), IDdip int not null references dipartimento(IDdip));
```

Creazione tabelle: Esempio

create table competenzeImpiegato(IDimpiegato int not null references impiegato (Idimpiegato), competenza varchar(15) not null, primary key (IDimpiegato, competenza));

create table cliente(IDcliente int not null auto_increment primary key, nome varchar(40), indirizzo varchar(100), nomeReferente varchar(80), telReferente char(12));

create table assegnamento(IDcliente int not null references cliente(IDcliente), IDimpiegato int not null references impiegato(IDimpiegato), dataLavoro date not null, oreEffettuate float, primary key (IDcliente, IDimpiegato, dataLavoro));

Creazione tabelle: Esempio

È opportuno rivedere i comandi uno a uno, a partire da:

drop database if exists impiegato;

Questo comando controlla se esiste già una base di dati impiegato e in tal caso la cancella, liberando, se volete, l'area di memoria. Questo comando non è strettamente necessario e potrebbe rivelarsi pericoloso; in questo esempio è inserito per essere sicuri che lo script funzioni anche se non è la prima volta che fate delle prove con una base di dati chiamata impiegato.

Fate attenzione che, se usate MySQL come programma inserito in un sistema ospite, il vostro ospite potrebbe aver disabilitato il comando drop database.

Creazione tabelle: Esempio

I comandi successivi sono dedicati alla creazione e alla selezione della base di dati:

create database impiegato;
use impiegato;

Ora potete creare le tabelle nella base di dati; cominciate con la tabella dipartimento, come segue:

create table dipartimento(IDdip int not null auto_increment primary key, nomeDip varchar(20));

Questa tabella ha due colonne, IDdip, che è anche la chiave primaria, e il nome del dipartimento. Per specificare le colonne della tabella basta dare una lista di definizioni di colonna, separate da una virgola, racchiusa tra parentesi tonde. Notate che gli attributi di una colonna non vanno separati da una virgola, ma solo le colonne stesse.
Questo comando è il primo comando SQL formato da più righe che vediamo.

Creazione tabelle: Esempio

Gli spazi non sono importanti in SQL, quindi si possono scrivere le istruzioni come si vuole. Di solito nei comandi CREATE si tende a mettere ogni oggetto in una riga diversa così da aumentare la leggibilità del codice. L'interprete SQL non esegue il comando finché non avete digitato il ; e premuto Invio. (Potete anche usare \g per terminare il comando, ma il punto e virgola è molto più comune.)

In questa tabella sono definite due colonne. La definizione di ciascuna inizia con il nome, seguito dall'informazione sul tipo di dato della colonna stessa. Guardate per prima la definizione della seconda colonna perché è più semplice da capire.

Creazione tabelle: Esempio

La definizione:

nome varchar(30)

dice che la colonna si chiama nome e che il suo tipo di dato è varchar(20). Il tipo ***varchar*** indica una stringa a lunghezza variabile, in questo caso può contenere fino a 30 caratteri. Si sarebbe potuto usare anche il tipo char, che indica una stringa a lunghezza fissa. Non esiste una particolare differenza nello scegliere varchar o char nel l'uso dei dati, ciò che è diverso è come i dati vengono salvati nella memoria del sistema. Un dato di tipo varchar(20) utilizza solo la memoria che serve per salvare il numero di caratteri di cui è composto, mentre un tipo char(20) occupa sempre lo spazio fisso di 20 caratteri anche se il dato inserito è più corto e quindi ne vorrebbe meno.

Creazione tabelle: Esempio

Osserviamo adesso la definizione della prima colonna. Ha questa forma:

IDdip int not null auto_increment primary key,

IDdip è il nome della colonna, ***int*** è il tipo di dato. Consiste in un numero univoco che verrà utilizzato per identificare ciascun dipartimento nella compagnia. Dopo il tipo di dato, sono presenti anche altre informazioni riguardo la colonna:
not null, cioè deve contenere un valore in ogni riga della tabella,
auto_increment cioè quando vengono inseriti dei dati nella tabella non si specifica un numero di dipartimento, MySQL assegna automaticamente il numero successivo in una sequenza di auto incremento dei numeri. (Questa caratteristica semplifica la vita!)
Infine viene detto che questa colonna rappresenta la ***primary key*** della tabella. Si può usare questo tipo di definizione ogni volta che la chiave primaria è composta da una sola colonna. Se fosse composta da più colonne occorrerebbe usare un approccio diverso...

Creazione tabelle: Esempio

Riferiamoci ora al secondo comando create table:

create table impiegato(IDimpiegato int not null auto_increment primary key, nome varchar(80), mansione varchar(30), IDdip int not null references dipartimento(IDdip));

In questo comando c'è soltanto un pezzo nuovo di sintassi. L'ultima colonna nella tabella impiegato è ID del dipartimento dove l'impiegato lavora. E una chiave esterna. La si definisce nella tabella aggiungendo il comando references:

IDdip int not null references dipartimento(IDdipartimento)

Il comando dice che IDdip nella tabella impiegato fa riferimento alla colonna IDdip della tabella dipartimento.

Creazione tabelle: Esempio

Adesso osserviamo il terzo comando create table:

create table competenzeImpiegato(IDimpiegato int not null references impiegato (Idimpiegato), competenza varchar(15) not null, primary key (IDimpiegato, competenza));

Anche in questa tabella c'è una chiave esterna, IDimpiegato. La cosa interessante di questa definizione di tabella è che si hanno due colonne nella chiave primaria. Potete osservare come prima vengano definite le due colonne, IDimpiegato e competenza, e poi venga definita la chiave primaria con il seguente comando:

primary key (IDimpiegato, competenza)

Le altre definizioni di tabella non contengono nuova sintassi, quindi non verranno trattate in dettaglio.

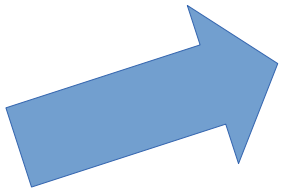
Creazione tabelle: Esempio

Possiamo comunque notare che sono stati usati un altro paio di tipi di dato: nella tabella assegnamento, il numero di oreEffettuate è definito float, o reale a virgola mobile (floating point), mentre dataLavoro è di tipo date (data). Vedremo meglio questi tipi di dato più avanti.

Possiamo ora controllare se le tabelle sono state salvate in modo corretto usando il comando:

show tables;

Dovrebbe comparire sul video



```
+-------------------------+
| Tables_in_impiegato     |
+-------------------------+
| assegnamento            |
| cliente                 |
| dipartimento            |
| impiegato               |
| competenzeImpiegato     |
+-------------------------+
```

Creazione tabelle: Esempio

Possiamo ottenere maggiori informazioni sulla struttura di ciascuna tabella usando il comando describe, ad esempio:

describe dipartimento;

Dovrebbe comparire a video la seguente schermata:

Field	Type	Collation	Null	Key	Default	Extra
IDdipartimento	int(11)	binary		PRI	NULL	auto_increment
nome	varchar(20)	latin1_swedish_ci	YES		NULL	

Controlliamo ora le altre tabelle.

Il comando create table

```
CREATE [TEMPORARY] TABLE [IF NOT EXIST] nome_tabella [( definizioni_create...)]
[opzioni sulla tabella] [comandi di selezione]
```

oppure

```
CREATE [TEMPORARY] TABLE [IF NOT EXIST] nome_tabella LIKE nome_altra_tabella;

definizioni_create:
    nome_colonna type [NOT NULL | NULL] [DEFAULT valore_standard] [AUTO_INCREMENT]
          [PRIMARY KEY] [definizione di chiave esterna (references)]
    (sottotype)
    oppure        PRIMARY KEY (nome_colonna_indice,...)
    oppure        KEY[nome_indice] (nome_colonna_indice,...)
    oppure        INDEX [nome_indice] (nome_colonna_indice,...)
    oppure        UNIQUE [INDEX] [nome_indice] (nome_colonna_indice,...)
    oppure        FULLTEXT [INDEX] [nome_indice] (nome_colonna_indice,...)
    oppure        [CONSTRAINT simbolo] FOREIGN KEY [nome_indice]
                  (nome_colonna_indice,...)
                  [definizione di chiave esterna (references)]
    oppure        CHECK (espressione)
```

Il comando create table

La parola chiave TEMPORARY indica che la tabella che si sta creando sarà visibile solo durante questa sessione di lavoro e verrà cancellata appena la connessione verrà chiusa.
Logicamente, potete usare l'opzione IF NOT EXISTS, per creare una tabella solo se non ne esiste già una con lo stesso nome nella base di dati.
Potete usare l'opzione LIKE nome_altra_tabella per creare una nuova tabella con lo stesso schema di nome_altra_tabella.
Dichiarate le colonne che volete, i loro tipi di dato, e qualsiasi altra informazione sulla struttura della tabella all'interno delle parentesi tonde. Il tipo di definizione più semplice per una colonna è dato dal nome della colonna seguito dal tipo di dato.

Le altre opzioni possibili per una colonna sono le seguenti:

- potete definire ciascuna colonna come NOT NULL oppure NULL, volendo dire che la colonna non può contenere dati nulli (NOT NULL) oppure che può contenerli (NULL);
- potete definire un valore standard per una colonna usando la parola chiave DEFAULT seguita dal valore standard che si vuole;

Il comando create table

Si usa la parola chiave AUTO_INCREMENT, per generare una sequenza di numeri. Il nuovo valore generato automaticamente sarà sempre maggiore di una unità rispetto al valore più grande presente in quel momento nella tabella. La prima riga inserita avrà valore 1 nella sequenza. È possibile avere solo una colonna AUTO_INCREMENT per tabella, e deve essere indicizzata. Notate come, nell'esempio precedente, non sia stato creato alcun indice manualmente. In ogni caso esistono degli indici creati automaticamente, cioè quelli per le colonne PRIMARY KEY, che sono le colonne AUTO_INCREMENT.
Si può indicare una particolare colonna come chiave primaria della tabella;
e dire che una colonna è una chiave esterna usando l'opzione REFERENCES come è stato fatto nell'esempio.

Il comando create table

Esattamente come per i nomi e i tipi di dato delle colonne, si possono definire altre informazioni della colonna in questa parte del comando CREATE TABLE:

- Si può definire una PRIMARY KEY a più colonne, come è stato fatto nell'esempio, scrivendo le parole chiave PRIMARY KEY seguite dai nomi delle colonne che formano la chiave. Questa scrittura può essere usata anche per definire una chiave formata da una sola colonna. Una colonna PRIMARY KEY è una colonna univoca, indicizzata e che non può contenere valori nulli;
- le parole INDEX e KEY sono sinonimi e specificano che la colonna in questione sarà indicizzata. Notate che queste colonne non devono per forza contenere valori unici in MySQL;

Il comando create table

- UNIQUE può essere usato per indicare che la colonna deve contenere valori non ripetuti. Anche la colonna segnata con UNIQUE sarà indicizzata;
- FULLTEXT viene usato per creare indici di testo in colonne di tipo TEXT, CHAR oppure VARCHAR. E possibile usare indici FULLTEXT solo nelle tabelle MyISAM
- l'opzione FOREIGN KEY vi permette di definire chiavi esterne negli stessi due modi con i quali sono state definite le chiavi primarie.

Il comando create table

Dopo aver chiuso le parentesi tonde, possoni usarsi opzioni per la tabella. Come l'opzione type che definisce il tipo di tabella, (default MyISAM).
Brevemente, questi sono i tipi di tabella possibili:

- MyISAM, il tipo standard, è molto veloce e supporta gli indici FULLTEXT. Sostituisce lo standard precedente ISAM (obsoleto) ed ha meno proprietà;
- InnoDB è un sistema di memorizzazione con proprietà ACIDe (Atomicy, Consistency, Isolation, Durability cioè Atomicità, Consistenza, Isolamento, Durata) che supporta transazioni, chiavi esterne e lock a livello di riga;
- BDB (Berkeley DB) è un motore di memorizzazione che supporta transazioni e lock a livello di pagina;
- le tabelle HEAP vengono salvate interamente in memoria locale e mai su disco, quindi sono molto veloci, ma hanno una dimensione limitata e non possono essere recuperate in caso di guasto del sistema;
- l'istruzione MERGE unisce più tabelle MyISAM con la stessa struttura, così da poterle interrogare come se fossero una sola tabella. Questo tipo permette di aggirare eventuali restrizioni del sistema operativo sulle dimensioni massime di un file, e, di conseguenza, di una tabella.

Esistono anche altre proprietà che possono essere specificate per una tabella. Non sono obbligatorie e servono per lo più per ragioni di ottimizzazione.

Tipi di colonna e dati in MySQL

In MySQL esistono tre tipi fondamentali di colonna: numerici, stringa o testo, e i tipi data e ora.
I tipi numerici vengono usati per salvare i numeri. Nell'esempio, sono stati usati i tipi **int** e **float**. Questi sono due sottotipi del tipo numerico: il tipo numerico esatto e quello approssimato.
I tipi numerici possono essere limitati da una dimensione di visualizzazione I e, per quelli a virgola mobile, dal numero di cifre decimali, D. Questi numeri vanno indicati dopo la dichiarazione di tipo; ad esempio:

stipendio decimal(10, 2)

verrà visualizzato con un massimo di 10 cifre delle quali 2 decimali.
Si può decidere di non indicare questi parametri, oppure definire solo il numero totale di cifre o dare sia il totale che il numero di cifre decimali.

I tipi numerici possono essere seguiti dalla parola chiave UNSIGNED e/o ZEROFILL. La prima indica che la colonna può contenere solo numeri positivi (zero incluso). ZEROFILL significa che il numero verrà visualizzato con degli zeri che precedono le cifre significative.

Tipi numerici in MySQL

NUMERIC o DECIMAL, indicano lo stesso tipo di dato numerico, la parola DECIMAL può essere abbreviata in DEC. Sono usati per memorizzare dei tipi esatti a virgola mobile e di solito vengono scelti per le valute. Hanno lo stesso intervallo di valori dei numeri a virgola mobile a doppia precisione (DOUBLE).
INTEGER, può essere abbreviato in INT. Indica l'intero standard a 4 byte con un intervallo di 2^{32} valori possibili. Esistono anche delle varianti:

Tipi Numerici Interi

Tipo	Intervallo di valori
TINYINT	-128 > 127
SMALLINT	-32768 > 32767
MEDIUMINT	-8388608 > 8388608
INT	-2147483648 > 2147483647
BIGINT	-9223372036854775808 > 9223372036854775807

Tipi Numerici NON Interi

Tipo
FLOAT(I,D)
DOUBLE(I,D)
DECIMAL(I,D)

Tipi carattere in MySQL

Tipi di dati Stringa

Tipo	Dimensioni massime
CHAR(n)	255 byte
VARCHAR(n)	255 byte
TINYTEXT	255 byte
TINYBLOB	255 byte
TEXT	65535 byte
BLOB	65535 byte
MEDIUMTEXT	1.6 Mb
MEDIUMBLOB	1.6 Mb
LONGTEXT	4.2 Mb
LONGBLOB	4.2 Mb

CHAR e VARCHAR, il primo ha un dimensione fissa: ad esmpio CHAR(10) avrà sempre una dimensione di 10 caratteri anche se ne inserissero 3 per un determinato record.
VARCHAR ha una dimensione legata all'effettiva grandezza del dato contenuto.
Se prendessimo l'esempio visto prima l'occupazione effettiva sarà dunque di 3 caratteri.

Tipi di date in MySQL

Tra queste sicuramente vanno menzionate le funzioni per l'ottenimento di data e ora corrente: NOW(), CURDATE() e CURTIME().

Alcune funzioni molto importanti riguardano la gestione del tempo, ossia aggiunta o sottrazione di intervalli di tempo (rispetto a date considerate).

Funzione	Formato della risposta
NOW()	AAAA-MM-GG HH:MM:SS
CURDATE()	AAAAA-MM-GG
CURTIME()	HH:MM:SS

Funzione	Utilizzo
DATE_ADD()	DATE_ADD(data, INTERVAL espressione tipo)
DATE_SUB()	DATE_SUBB(data, INTERVAL espressione tipo)
PERIOD_ADD()	PERIOD_ADD(Periodo, Mesi) il Periodo va espresso informato AAAAMM oppure AAMM
PERIOD_SUBB()	PERIOD_SUB(Periodo1, Periodo2) il Periodo va espresso informato AAAAMM oppure AAMM

Esempio1: vogliamo aggiungere 15 giorni alla data corrente:
DATE_ADD(CURRDATE(), INTERVAL 15 days);

Esempio2: vogliamo sottrarre 2 mesi dalla data specificata
DATE_SUB('2005-01-23', INTERVAL 2 months);

Creazione indici in MySQL

Di solito, gli indici vengono creati quando vengono create le tabelle. Tutte le colonne dichiarate come PRIMARY KEY, KEY, UNIQUE o INDEX vengono automaticamente indicizzate.
A volte può capitare che ci si accorga che molte interrogazioni coinvolgono una colonna non indicizzata. In questo caso si può aggiungere un indice usando il comando CREATE INDEX.
Una cosa abbastanza interessante è che il comando CREATE INDEX provoca sempre l'esecuzione di un comando ALTER TABLE per poter essere eseguito a sua volta. Esistono altri casi oltre a questo in cui usare il comando ALTER TABLE.
Ad esempio, potete aggiungere un indice alla tabella impiegato nel modo seguente:

create index nome on impiegato(nome);

Il comando crea un indice chiamato nome sui campo nome della tabella impiegato.

Ancora indici in MySQL

Il comando non ha molte opzioni possibili. Si può far precedere la parola index con UNIQUE per rinforzare il vincolo di unicità. Oppure premettere la parola chiave FULLTEXT se si vuole creare un indice di testo su una tabella MyISAM.
L'unica vera opzione è quella di limitare gli indici sui tipi CHAR e VARCHAR per indicizzare solo i primi caratteri di ciascun campo, specificando il numero di caratteri dell'indice tra parentesi tonde subito dopo il nome della colonna da indicizzare.
Ad esempio:

create index parte_nome on impiegato(nome(5))

Una ragione per usare questa opzione è che gli indici sui campi testuali sono molto meno efficienti di quelli sui campi numerici, e indicizzare solo i primi caratteri migliora le prestazioni.

Eliminazione DB, tabelle, indici

Adesso che si sanno creare basi di dati, tabelle e indici, è utile sapere come eliminarli. La parola chiave è DROP.
Si può cancellare un'intera base di dati, compreso tutto il suo contenuto, con il comando seguente:

drop database impiegato;

si può aggiungere l'opzione IF EXISTS prima del nome della base di dati. Questa è esattamente la versione del comando DROP DATABASE usata nell'esempio precedente.
Si può cancellare una singola tabella col comando DROP TABLE:

drop table assegnamento;

Eliminazione DB, tabelle, indici

La forma generale del comando DROP TABLE è la seguente:

DROP [TABLE [EXISTS] nome_tabella [nome_tabella,...]

Si può specificare la parola chiave TEMPORARY per eliminare le tabelle temporanee. Possono cancellarsi più tabelle in una volta sola elencando i loro nomi separati da una virgola. L'opzione IF EXISTS funziona nello stesso modo che in DROP DATABASE.
Si può eliminare un indice con il comando DROP INDEX, ad esempio:

drop index parte_nome on impiegato;

Come si vede, è necessario specificare la tabella che contiene l'indice da cancellare.

Modifica una tabella esistente

Per modificare la struttura di una tabella esistente si usa il comando ALTER TABLE.

Ad es. per creare un indice chiamato nome nella tabella impiegato nel seguente modo:

alter table impiegato add index nome(nome);

Dal momento che il comando ALTER TABLE è molto flessibile, possiede molte possibili opzioni.

```
modifica_specifica:
        ADD [COLUMN] crea_definizione [FIRST | AFTER nome_colonna ]
    or  ADD [COLUMN] (crea_definizione, crea_definizione,...)
    or  ADD INDEX [nome_indice] (nome_ind_colonna,...)
    or  ADD PRIMARY KEY (nome_ind_colonna,...)
    or  ADD UNIQUE [nome_indice] (nome_ind_colonna,...)
    or  ADD FULLTEXT [nome_indice] (nome_ind_colonna,...)
    or  ADD [CONSTRAINT symbol] FOREIGN KEY [nome_indice] (nome_ind_colonna,...)
            [definizione di chiave esterna (references)]
    or  ALTER [COLUMN] nome_colonna {SET DEFAULT literal | DROP DEFAULT}
    or  CHANGE [COLUMN] vecchio_nome_colonna create_definition
            [FIRST | AFTER nome_colonna]
    or  MODIFY [COLUMN] crea_definizione [FIRST | AFTER nome_colonna]
    or  DROP [COLUMN] nome_colonna
    or  DROP PRIMARY KEY
    or  DROP INDEX nome_indice
    or  DISABLE KEYS
    or  ENABLE KEYS
    or  RENAME [TO] nuovo_nome_tabella
    or  ORDER BY nome_colonna
    or  opzioni_tabella
```

Modifica una tabella esistente

Molte di queste opzioni corrispondono alle opzioni del comando CREATE TABLE, come ad esempio ADD PRIMARY KEY.
CHANGE e MODIFY sono praticamente uguali: permettono di cambiare la definizione di una colonna o la sua posizione nella tabella.
DROP COLUMN elimina una colonna dalla tabella, mentre DROP PRIMARY KEY e DROP INDEX eliminano solamente l'indice associato a quella colonna.
DISABLE KEYS dice a MySQL, solo per tabelle MyISAM, di non aggiornare gli indici mentre ENABLE KEYS ripristina la possibilità di fare queste modifiche.
RENAME permette di cambiare il nome di una tabella.
ORDER BY sistema le righe della tabella appena modificata in un ordine particolare (non permanente perchè i dati nella cambiano nel tempo).
L'opzione **opzioni tabella** permette di specificare le stesse opzioni che erano permesse alla fine del comando CREATE TABLE, come spiegato all'inizio.

Esercizi

1. Scrivere i comandi SQL necessari per creare una base di dati (gestioneOrdini) col seguente schema:

   cliente(IDcliente, nomeCliente, indirizzoCliente)

   ordine(IDordine, dataOrdine, IDcliente)

   prodottoInOrdine(IDordine, IDprodotto, totProdotto)

   prodotto (IDprodotto, nomeProdotto)

   Fare tutte le ipotesi che si vogliono sui tipi di dato. Provare il codice in MySQL e controllare le tabelle risultanti con i comandi SHOW e DESCRIBE

1. Aggiungere un campo nota, di tipo testo, a ciascun ordine della tabella ordine. Usare il comando ALTER TABLE per farlo, e controllare il risultato con il comando DESCRIBE
2. Eliminare la base dati gestioneOrdini

Soluzione

create database gestioneOrdini;

use gestioneOrdini;

create table clienti(IDcliente int not null auto_increment primary key, nomeCliente varchar(20), indirizzoCliente varchar(80));

create table ordini(IDordine int not null auto_increment primary key, dataOrdine date, IDcliente int not null references customer(customerID));

create table prodotti(IDprodotto int not null auto_increment primary key, nomeProdotto varchar(20));

create table prodottoInOrdine(IDordine int not null references ordini(IDordine), IDprodotto int not null references prodotti(IDprodotto), totProdotto int, primary key (IDordine, IDprodotto));

Soluzione

Aggiungere un campo nota, di tipo testo, a ciascun ordine della tabella orders. Usare il comando ALTER TABLE per farlo, e controllare il risultato con il comando DESCRIBE

alter table ordini add column nota text;

Eliminare la base dati

drop database gestioneOrdini;